AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Original) A system that facilitates communicating between managed and unmanaged code, comprising:

a first component that is one of the managed and unmanaged code; and

a caller associated with the first component, the caller invoking an object related to a second component, the second component being one of the managed and unmanaged code, the caller including an in-lined stub that facilitates communications between the objects.

2. (Original) The system of claim 1, the in-lined stub including a call and return pair to facilitate communications between the objects.

3. (Original) The system of claim 1, further comprising a stack marker that is hoisted from within a code loop associated with the caller to facilitate code execution performance during communications between the objects.

4. (Original) The system of claim 1, the caller further comprising transition code to synchronize execution between the objects.

5. (Original) The system of claim 1, the caller further comprising one or more flags to synchronize execution between the objects.

6. (Original) The system of claim 5, the one or more flags utilized to synchronize code execution with a garbage collector.

7. (Original) The system of claim 6, the one or more flags utilized to suspend return operations from the unmanaged code until operations associated with the garbage collector have completed.

8. (Original) The system of claim 1, the caller further comprising security attribute code to insulate the objects from at least one of code and security implementation details.

9. (Original) The system of claim 1, the caller further comprising calling convention code to at least one of organize arguments and an execution stack according to the convention of the unmanaged code.

10. (Original) The system of claim 9, the calling convention code utilized to interpret one or more return values from the unmanaged code.

11. (Original) The system of claim 1, the caller including at least one of an in-lined marshalling code and an external marshalling code to transfer data between the objects.

12. (Original) The system of claim 1, the caller further comprising an extensibility component to facilitate generalized communications between the objects.

13. (Original) The system of claim 12, the extensibility component including a function pointer that includes one or more functions as arguments.

14. (Original) A method to facilitate communications between managed and unmanaged code, comprising:

incorporating external stub code functionality within a calling function; and

providing a call and return within the caller to facilitate communications between the managed and unmanaged code.

15. (Original) The method of claim 14, further comprising hoisting a marker from a loop associated with the caller to facilitate code execution performance during communications between the managed and unmanaged code.

16. (Original) The method of claim 14, further comprising synchronizing execution between the managed and unmanaged code.

17. (Original) The method of claim 16, further comprising synchronizing code execution with a garbage collector.

18. (Original) The method of claim 17, further comprising suspending operations from the unmanaged code until operations associated with the garbage collector have completed.

19. (Original) The method of claim 14, further comprising utilizing security attributes to insulate the managed and unmanaged code from at least one of code and security implementation details.

20. (Original) The method of claim 14, further comprising at least one of organizing arguments and an execution stack according to the format of the unmanaged code.

21. (Original) The method of claim 14, transferring data between the managed and unmanaged code *via* at least one of an in-lined marshalling code and an external marshalling code.

22. (Original) The method of claim 14, further comprising generalizing a function to facilitate communications between the managed and unmanaged code.

23. (Original) The method of claim 22, further comprising utilizing a function pointer that points to one or more functions as arguments to facilitate communications between the managed and unmanaged code.

24. (Original) A computer-readable medium having computer-executable instructions for performing the acts of claim 14.

25. (Original) A system to facilitate communications between disparate objects, comprising:

means for incorporating external stub code functionality within a caller; and

means for interfacing with the caller to facilitate communications between the disparate objects.

26. (Currently Amended) The system of claim 25, further comprising means for hoisting a marker from a loop associated with the caller to facilitate code execution performance during communications between the disparate objects.

27. (Original) A computer-readable medium having stored thereon, a caller to execute a remote object, comprising:

a first data set providing call and return interface capabilities within the caller;

a second data set providing execution stack instructions for the caller; and

a third data set providing hoisting capabilities within the caller, the interface, the stack instructions and the hoisting capabilities enabling execution of the remote object.

28. (Original) A signal to communicate between managed and unmanaged code, comprising:

a signal to communicate data between the managed and unmanaged code; and

a caller associated with the managed code, the caller invoking an object related to the unmanaged code *via* the signal, the caller including an in-lined stub that facilitates communications between the managed and the unmanaged code.